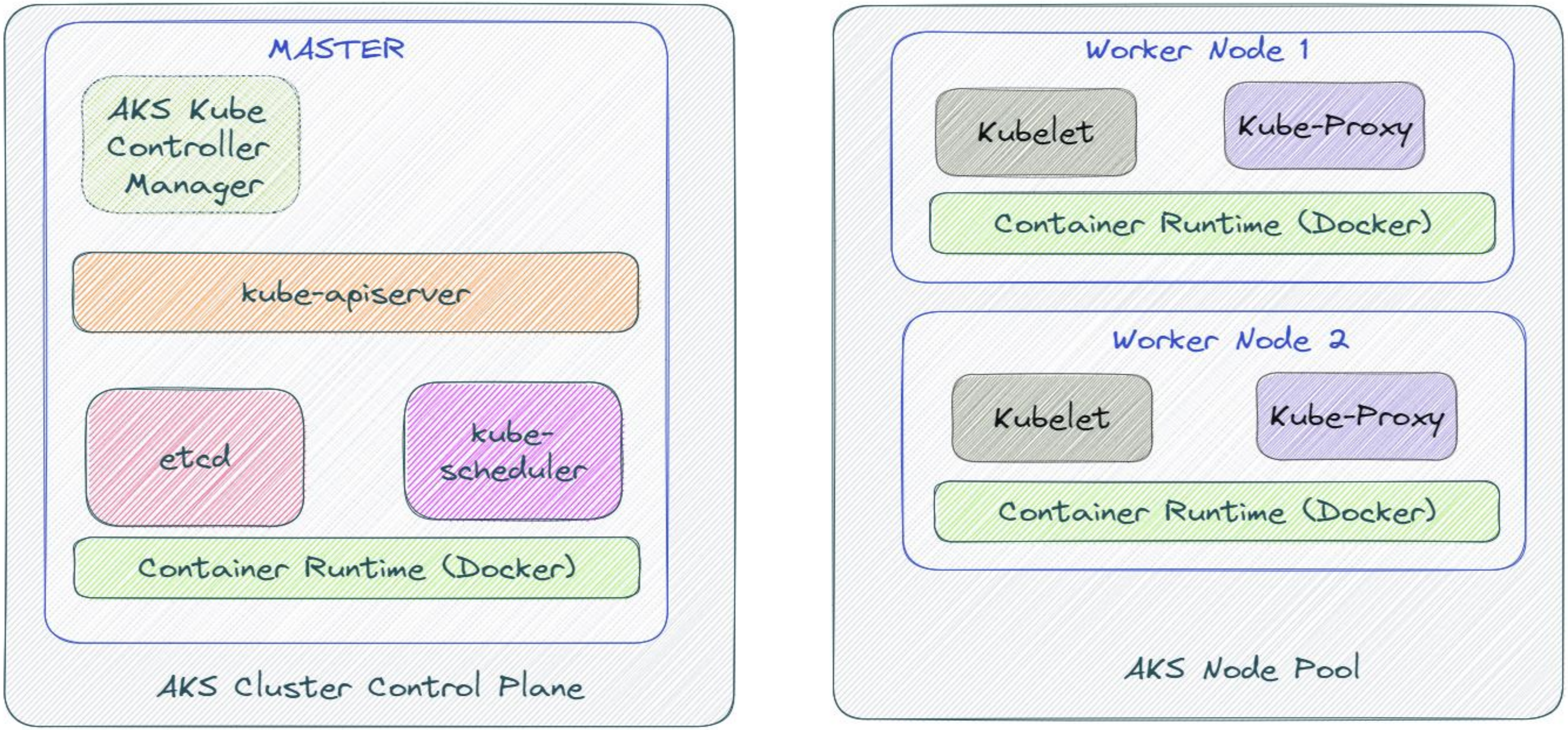
# AZURE AKS
# Create Cluster
# Introduction

# AKS - Introduction

- AKS – Azure Kubernetes Service
- AKS is highly available, secure and fully managed Service
- When compared to other cloud providers, AKS is the one which is available in highest number of regions
- Will be able to run any type of workloads
  - Windows based applications like .Net Apps
  - Linux supported applications like Java
  - IOT device deployment and management on demand
  - Machine Learning Model training with AKS
- Able to run in Hybrid Platforms
  - Azure StacK HCI
  - Windows Servers with Linux Distros
  - Planing for Vmware Platform
- Able to use Azure services without additional infra and admin effort
  - You can deploy and integrate azure services with your AKS easly
  - Azure Storage, Azure Key Vault, Azure Devops, Azure LB, etc.



AKS Cluster Control Plane

MASTER

AKS Kube Controller Manager

kube-apiserver

etcd

kube-scheduler

Container Runtime (Docker)

AKS Node Pool

Worker Node 1

Kubelet

Kube-Proxy

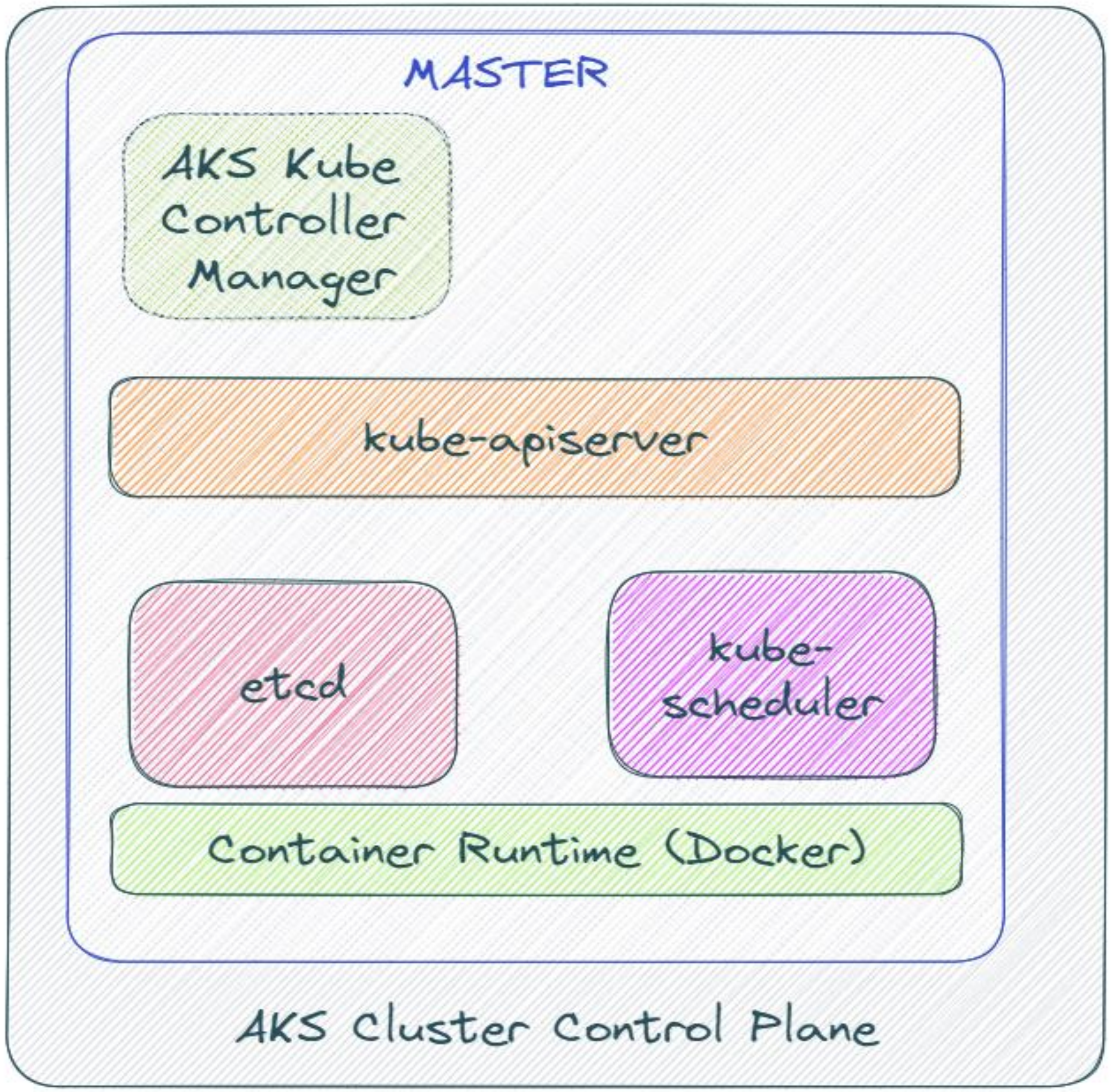
Container Runtime (Docker)

Worker Node 2

Kubelet

Kube-Proxy

Container Runtime (Docker)



MASTER

AKS Kube Controller Manager

kube-apiserver

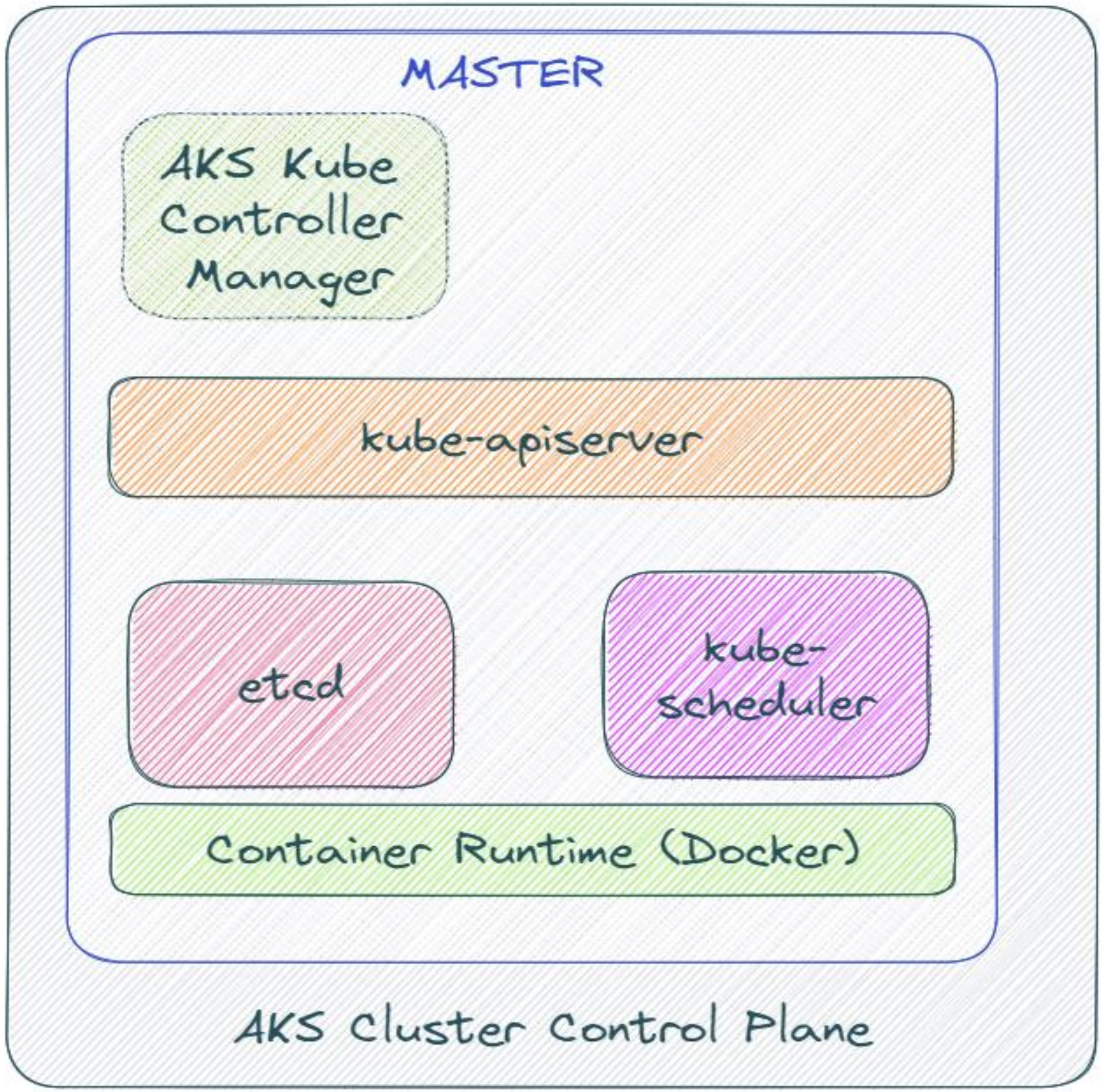
etcd

kube-scheduler

Container Runtime (Docker)

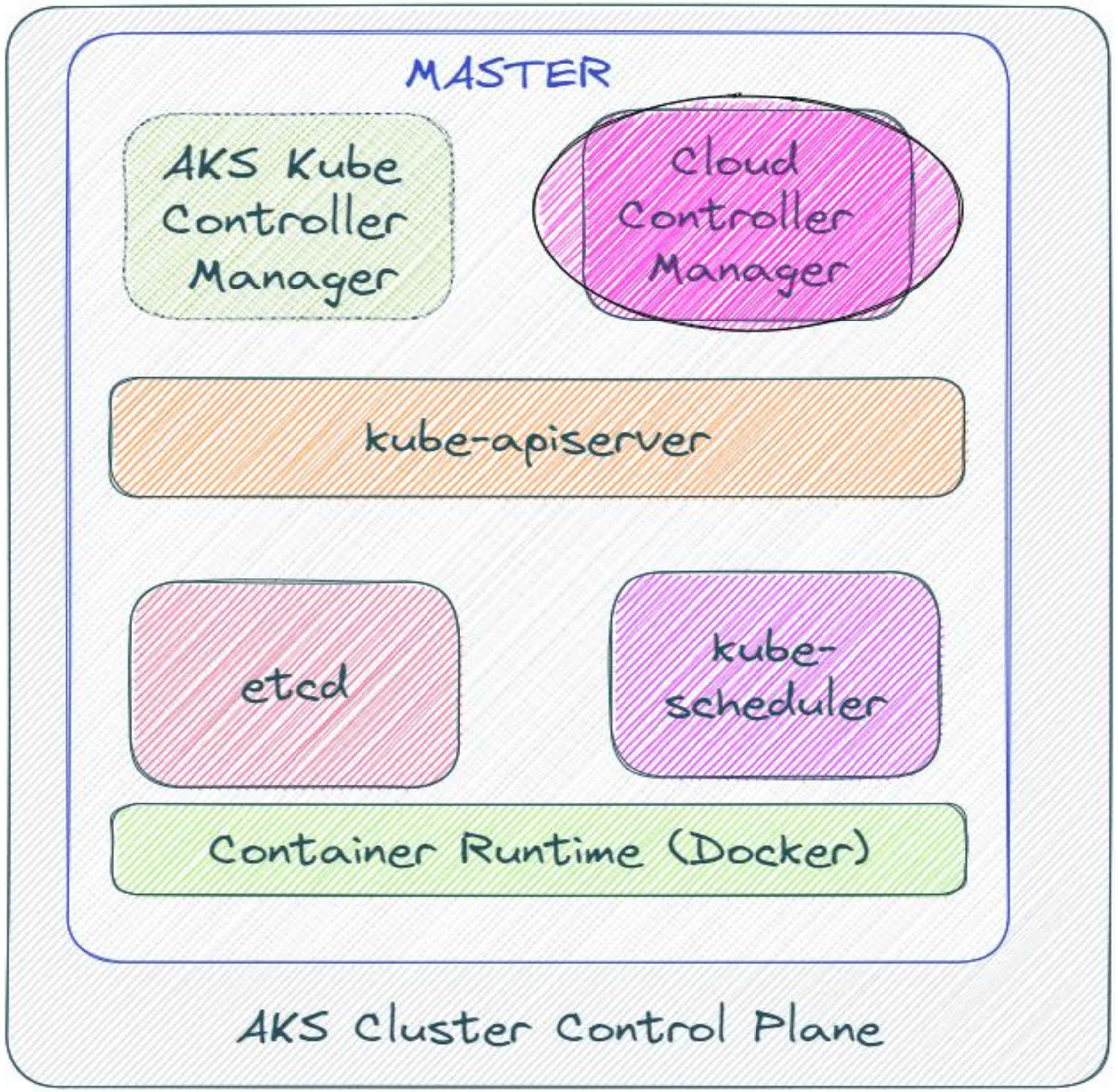
AKS Cluster Control Plane

- kube-apiserver
  - It acts as front end for the Kubernetes control plane. It exposes the Kubernetes API
  - Command line tools (like kubectl), Users and even Master components (scheduler, controller manager, etcd) and Worker node components like (Kubelet) everything talk with API Server.
- etcd
  - Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
  - It stores all the masters and worker node information.
- kube-scheduler
  - Scheduler is responsible for distributing containers across multiple nodes.
  - It watches for newly created Pods with no assigned node, and selects a node for them to run on.

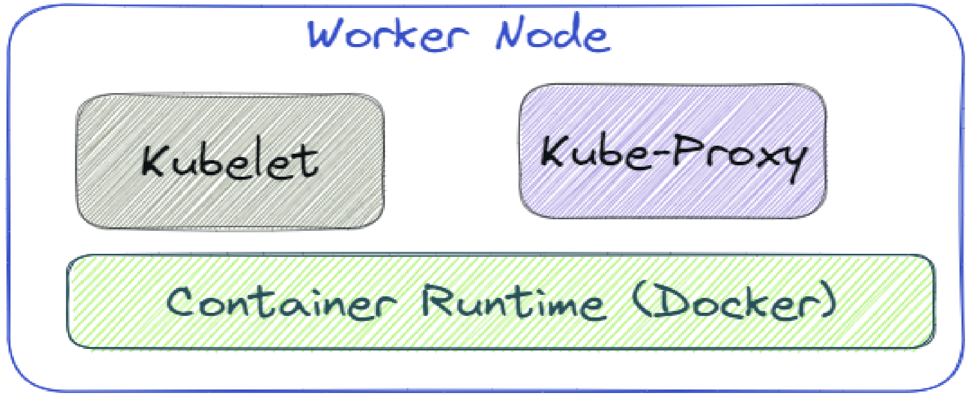

MASTER
AKS Kube Controller Manager
kube-apiserver
etcd
kube-scheduler
Container Runtime (Docker)
AKS Cluster Control Plane

○ kube-controller-manager

- Controllers are responsible for noticing and responding when nodes, containers or endpoints go down. They make decisions to bring up new containers in such cases.
- Node Controller: Responsible for noticing and responding when nodes go down.
- Replication Controller: Responsible for maintaining the correct number of pods for every replication controller object in the system.
- Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods)
- Service Account & Token Controller: Creates default accounts and API Access for new namespaces.


MASTER
AKS Kube Controller Manager
Cloud Controller Manager
kube-apiserver
etcd
kube-scheduler
Container Runtime (Docker)
AKS Cluster Control Plane

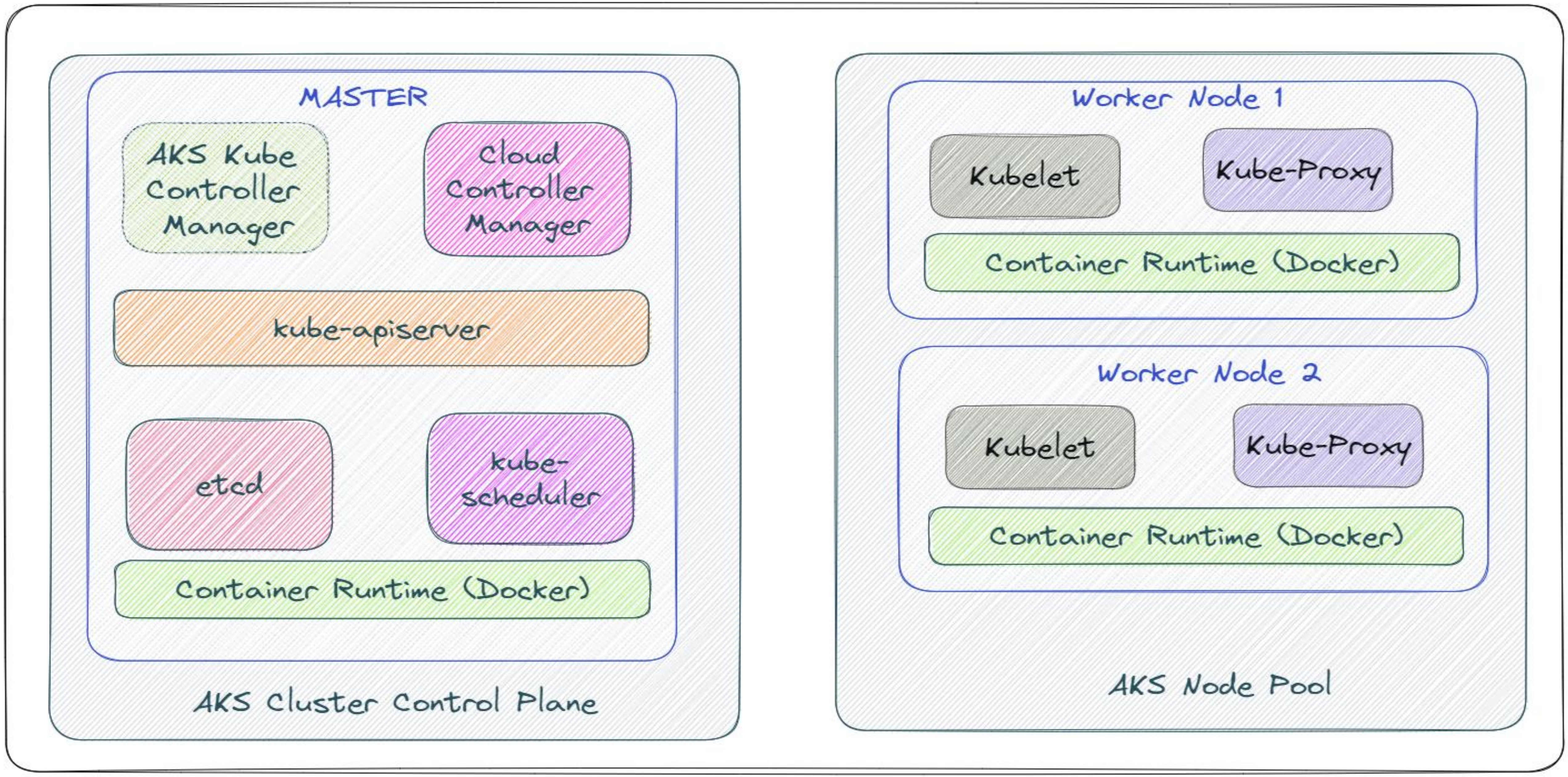
- cloud-controller-manager
  - A Kubernetes control plane component that embeds cloud-specific control logic.
  - It only runs controllers that are specific to your cloud provider.
  - On-Premise Kubernetes clusters will not have this component.
  - Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding.
  - Route controller: For setting up routes in the underlying cloud infrastructure.
  - Service controller: For creating, updating and deleting cloud provider load balancer.



- Container Runtime
  - Container Runtime is the underlying software where we run all these
  - We are using Docker, but we have other runtime options like rkt, container-d etc.

- Kubelet
  - Kubelet is the agent that runs on every node in the cluster
  - This agent is responsible for making sure that containers are running in a Pod on a node.

- Kube-Proxy
  - It is a network proxy that runs on each node in your cluster.
  - It maintains network rules on nodes
  - In short, these network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

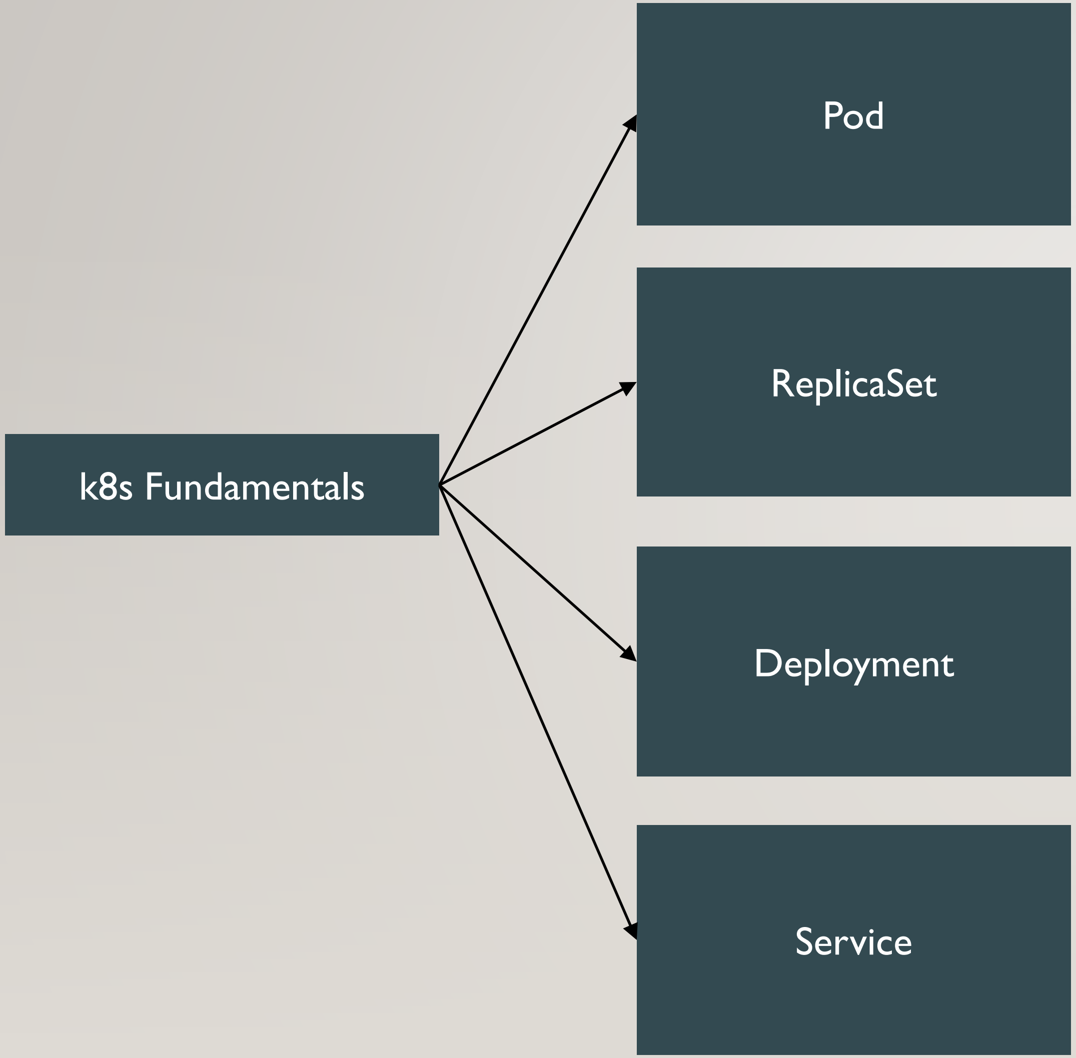

MASTER
AKS Kube Controller Manager
Cloud Controller Manager
kube-apiserver
etcd
kube-scheduler
Container Runtime (Docker)
AKS Cluster Control Plane
Worker Node 1
Kubelet
Kube-Proxy
Container Runtime (Docker)
Worker Node 2
Kubelet
Kube-Proxy
Container Runtime (Docker)
AKS Node Pool

# Kubernetes

# Fundamentals

## Pod, ReplicaSet, Deployment & Service

# KUBERNETES - FUNDAMENTALS



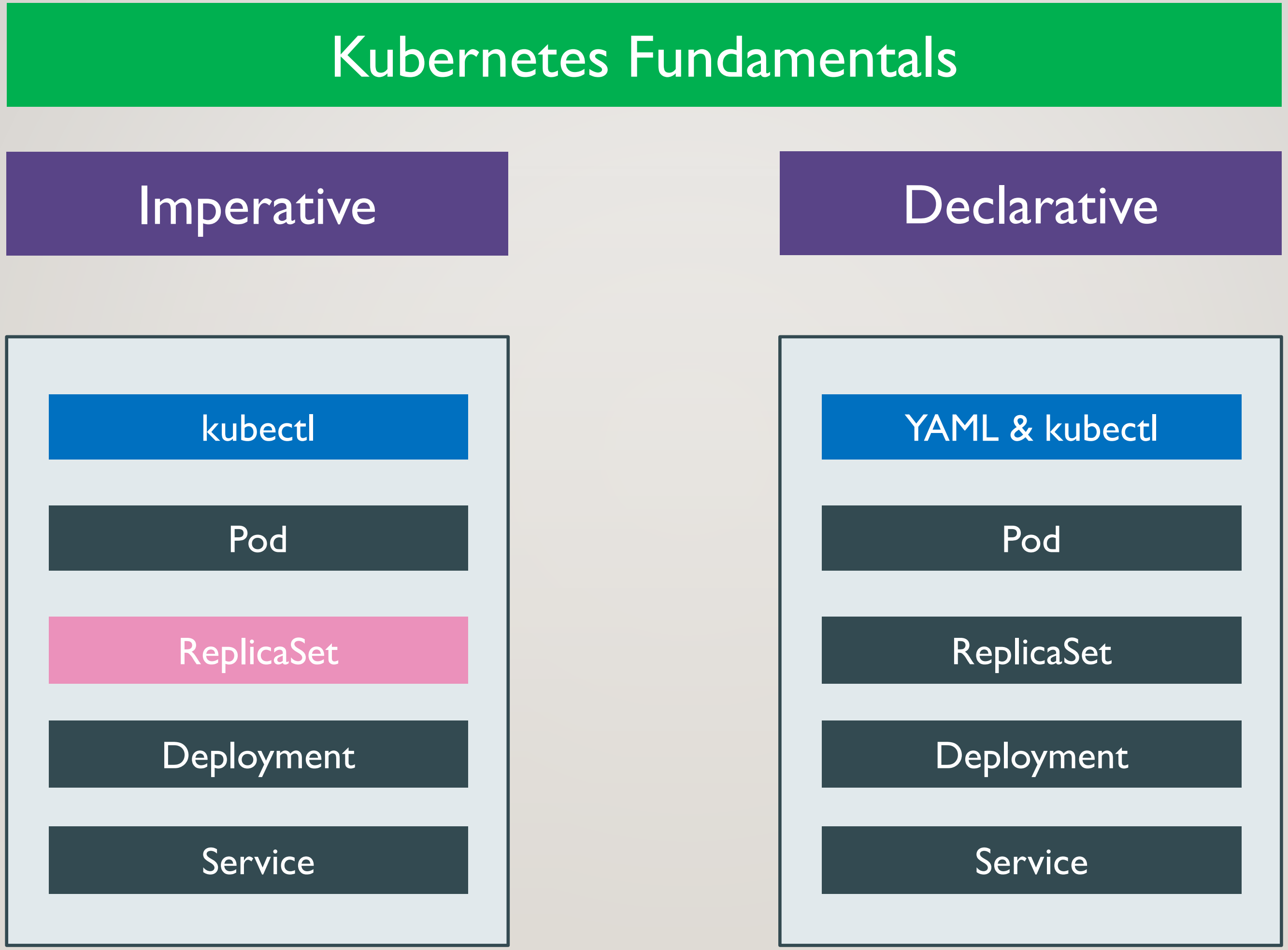
A POD is a single instance of an Application.
A POD is the smallest object, that you can create in Kubernetes.

A ReplicaSet will maintain a stable set of replica Pods running at any given time.
In short, it is often used to guarantee the availability of a specified number of identical Pods

A Deployment runs multiple replicas of your application and automatically replaces any instances that fail or become unresponsive. Rollout & rollback changes to applications. Deployments are well-suited for stateless applications.

A service is an abstraction for pods, providing a stable, so called virtual IP (VIP) address.
In simple terms, service sits Infront of a POD and acts as a load balancer.

# KUBERNETES - IMPERATIVE & DECLARATIVE



Kubernetes Fundamentals

Imperative

- kubectl
- Pod
- ReplicaSet
- Deployment
- Service

Declarative

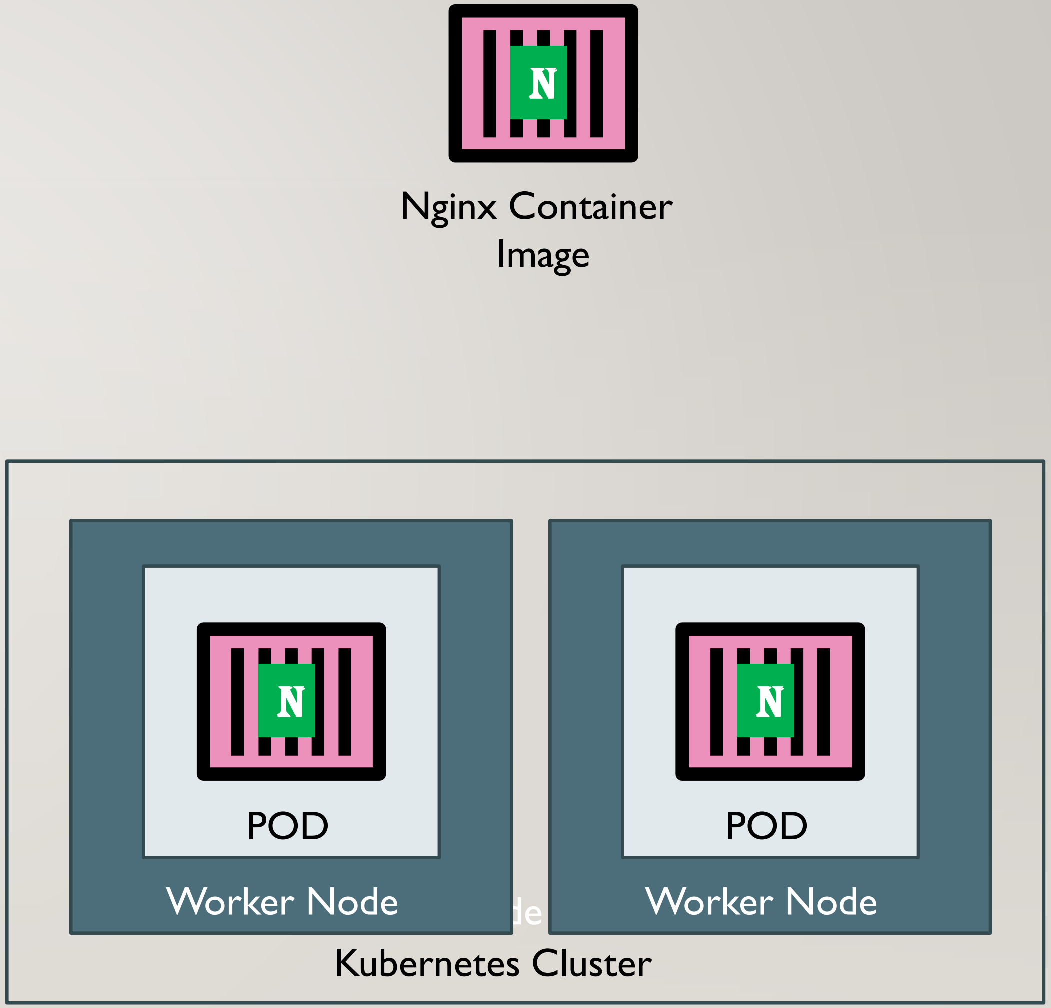
- YAML & kubectl
- Pod
- ReplicaSet
- Deployment
- Service

# Kubernetes

## POD

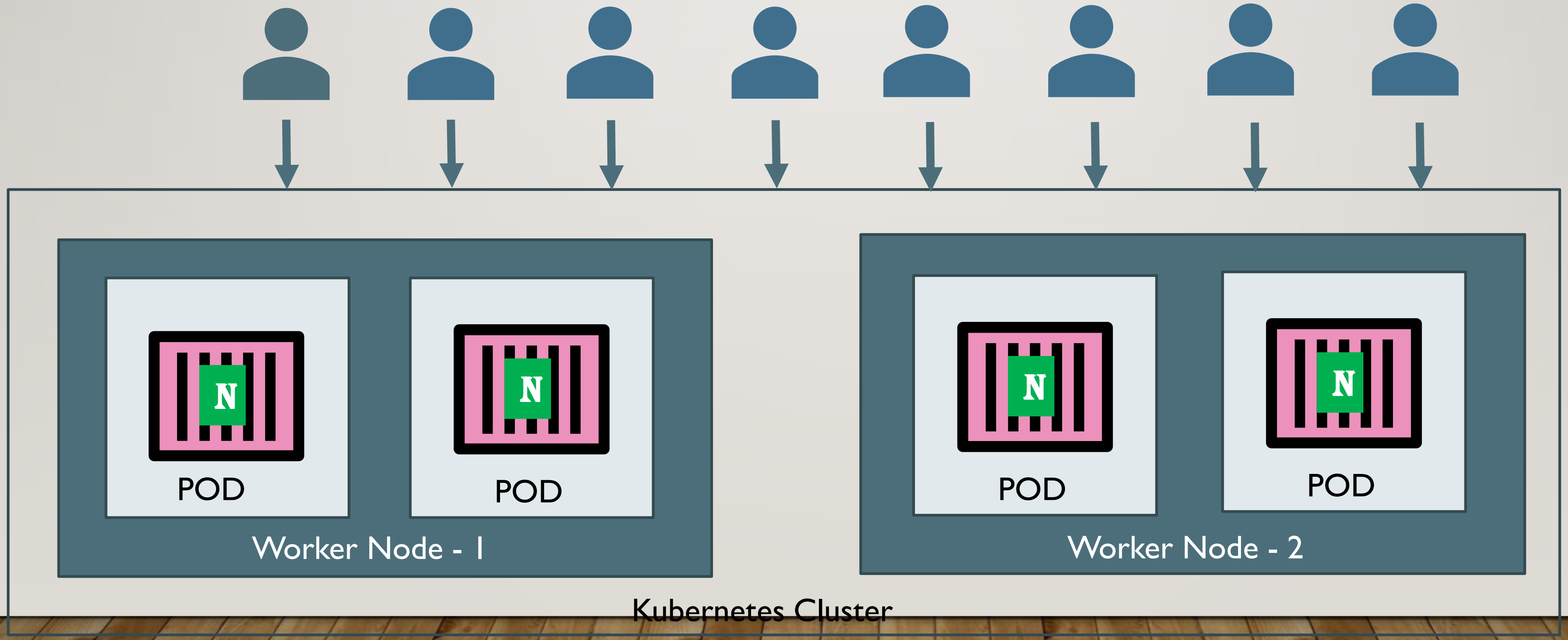
# KUBERNETES - POD

- With Kubernetes our core goal will be to deploy our applications in the form of containers on worker nodes in a k8s cluster.
- Kubernetes does not deploy containers directly on the worker nodes.
- Container is encapsulated in to a Kubernetes Object named POD.
- A POD is a single instance of an application.
- A POD is the smallest object that we can create in Kubernetes.



# KUBERNETES - POD

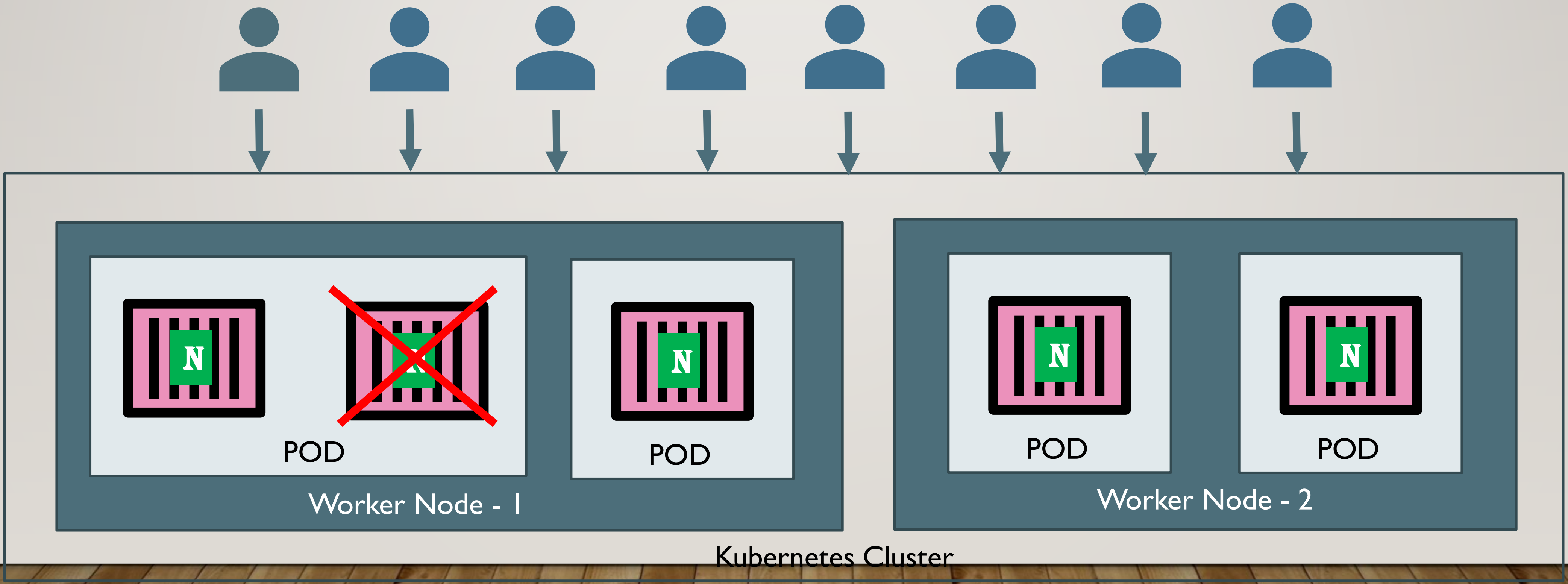
- PODs generally have one to one relationship with containers.
- To scale up we create new POD and to scale down we delete the POD.



Alper Can

# KUBERNETES – PODS

- We cannot have multiple containers of same kind in a single POD.
- Example: Two NGINX containers in single POD serving same purpose is not recommended.

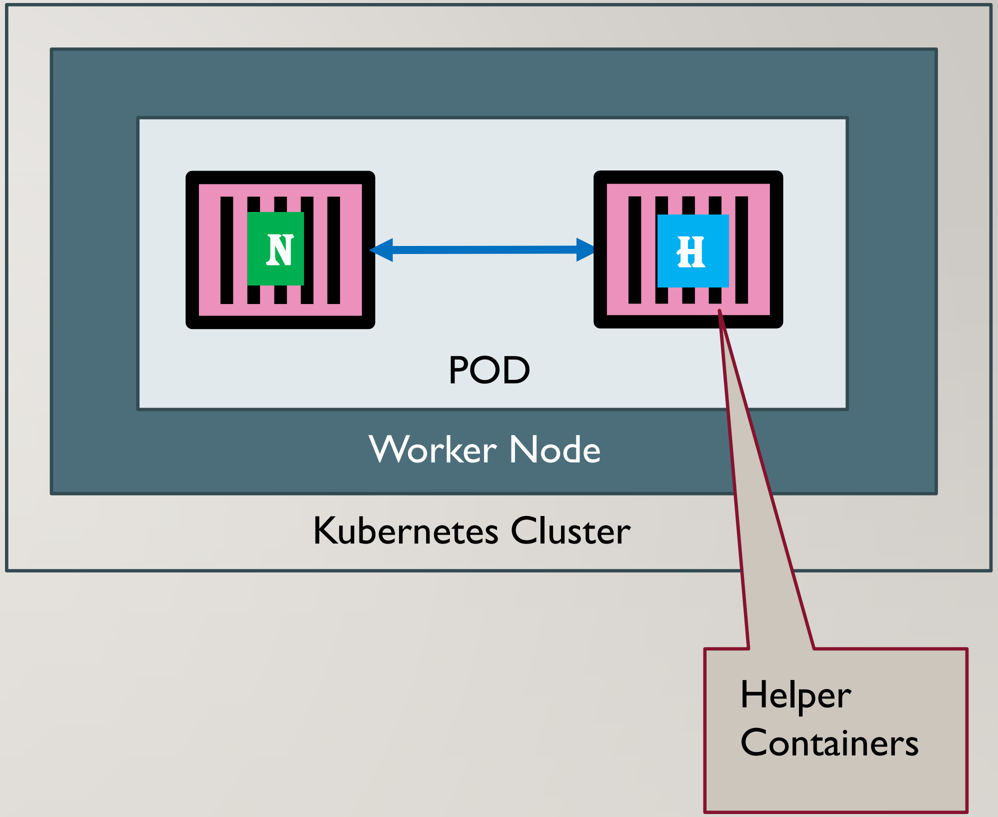


Alper Can

# KUBERNETES – MULTI-CONTAINER PODS

- We can have multiple containers in a single POD, provided they are not of same kind.
- Helper Containers (Side-car)
  - Data Pullers: Pull data required by Main Container
  - Data pushers: Push data by collecting from main container (logs)
  - Proxies: Writes static data to html files using Helper container and Reads using Main Container.
- Communication
  - The two containers can easily communicate with each other easily as they share same network space.
  - They can also easily share same storage space.
- Multi-Container Pods is a rare use-case and we will try to focus on core fundamentals.

Node



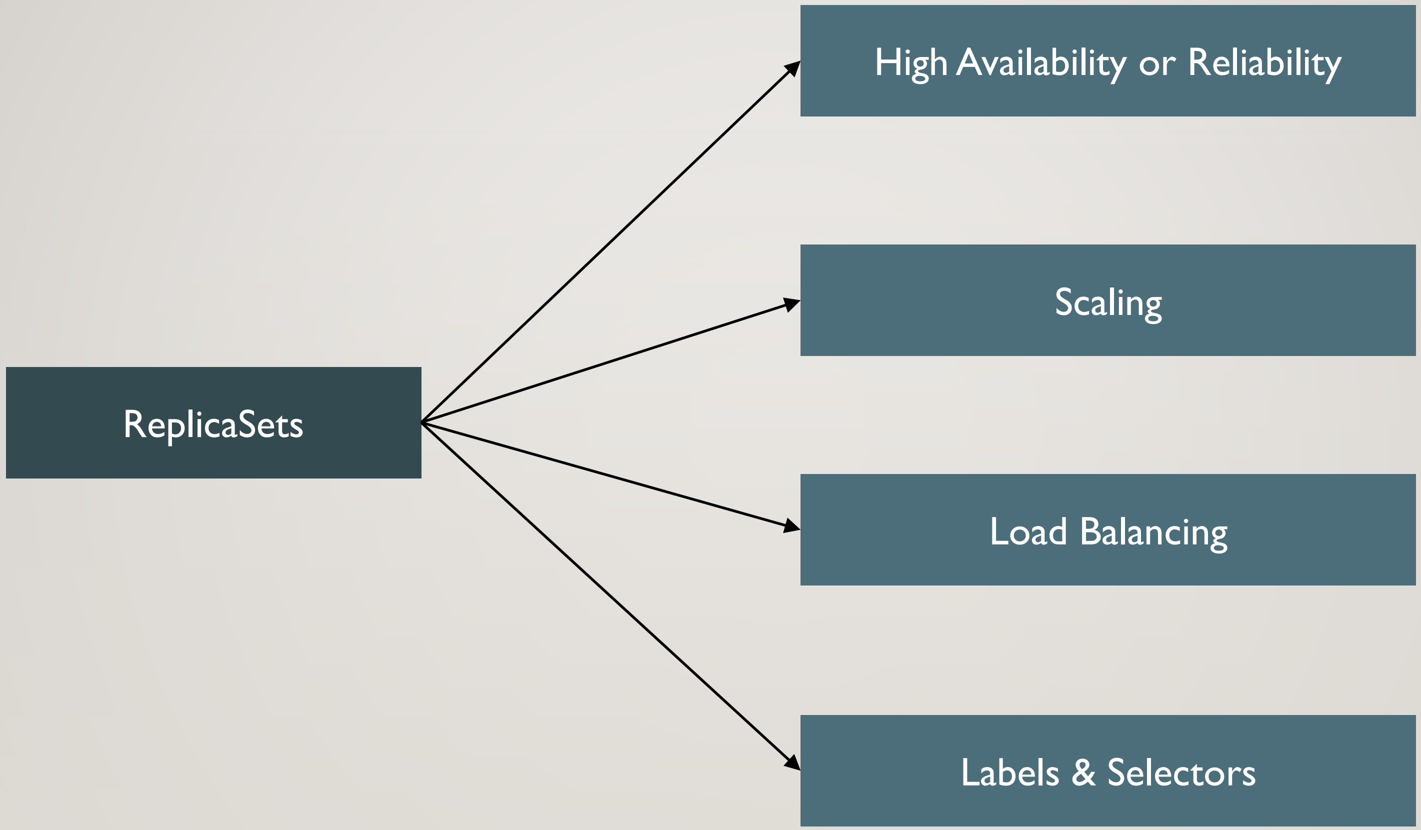
Alper Can

# Kubernetes

# ReplicaSets

Alper Can

# KUBERNETES - REPLICASETS



ReplicaSets

High Availability or Reliability

Scaling

Load Balancing

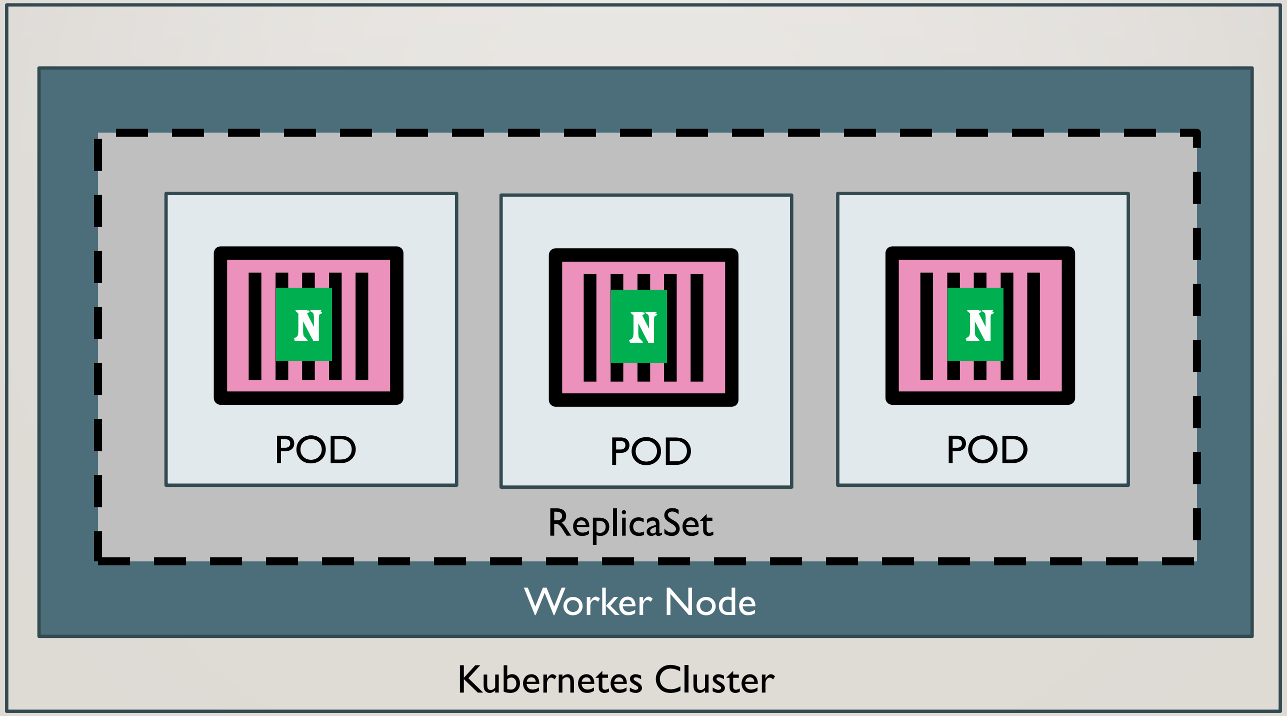
Labels & Selectors

# KUBERNETES – REPLICASET

- A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

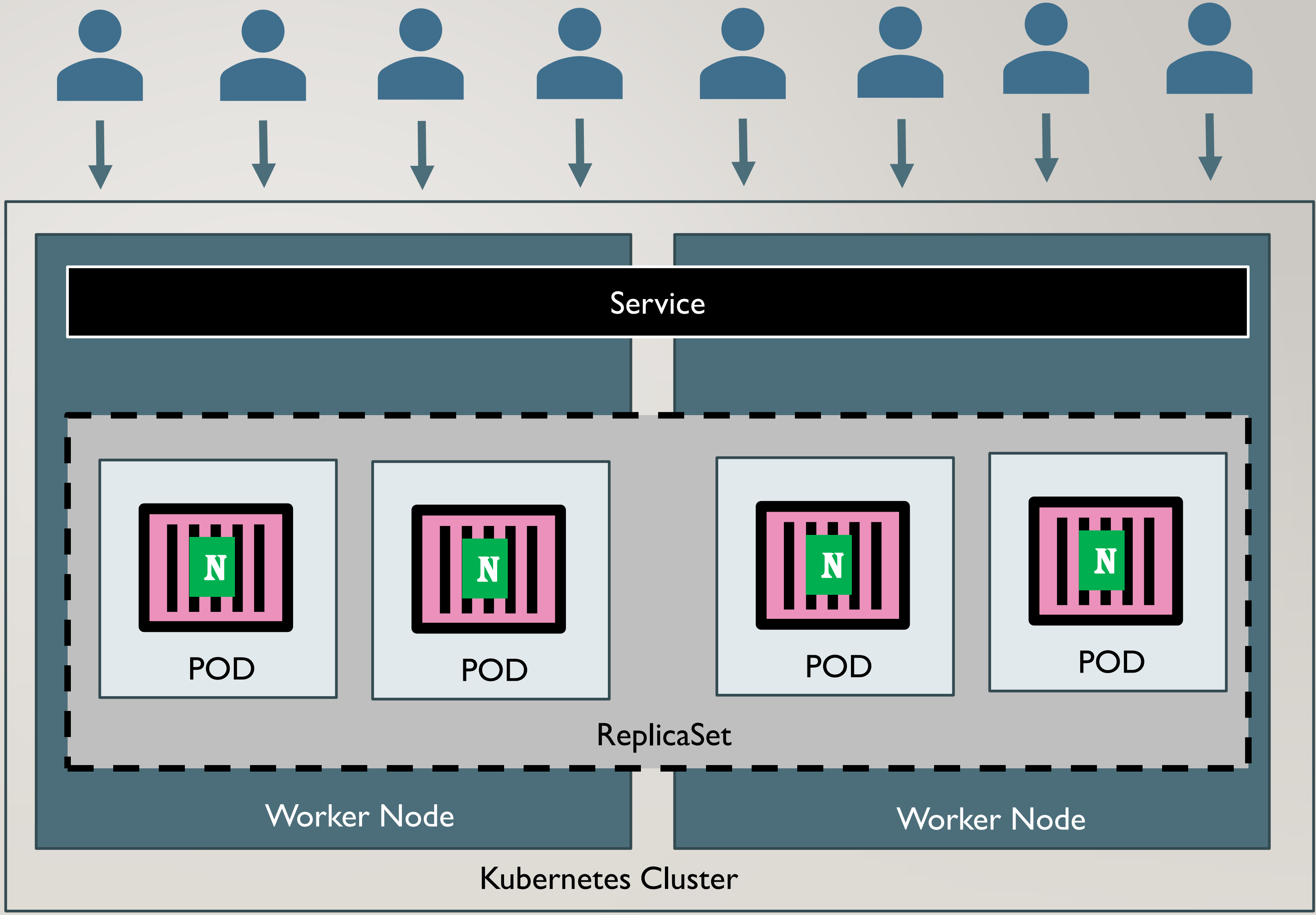
- If our application crashes (any pod dies), replicaset will recreate the pod immediately to ensure the configured number of pods running at any given time.

Reliability
Or
High Availability



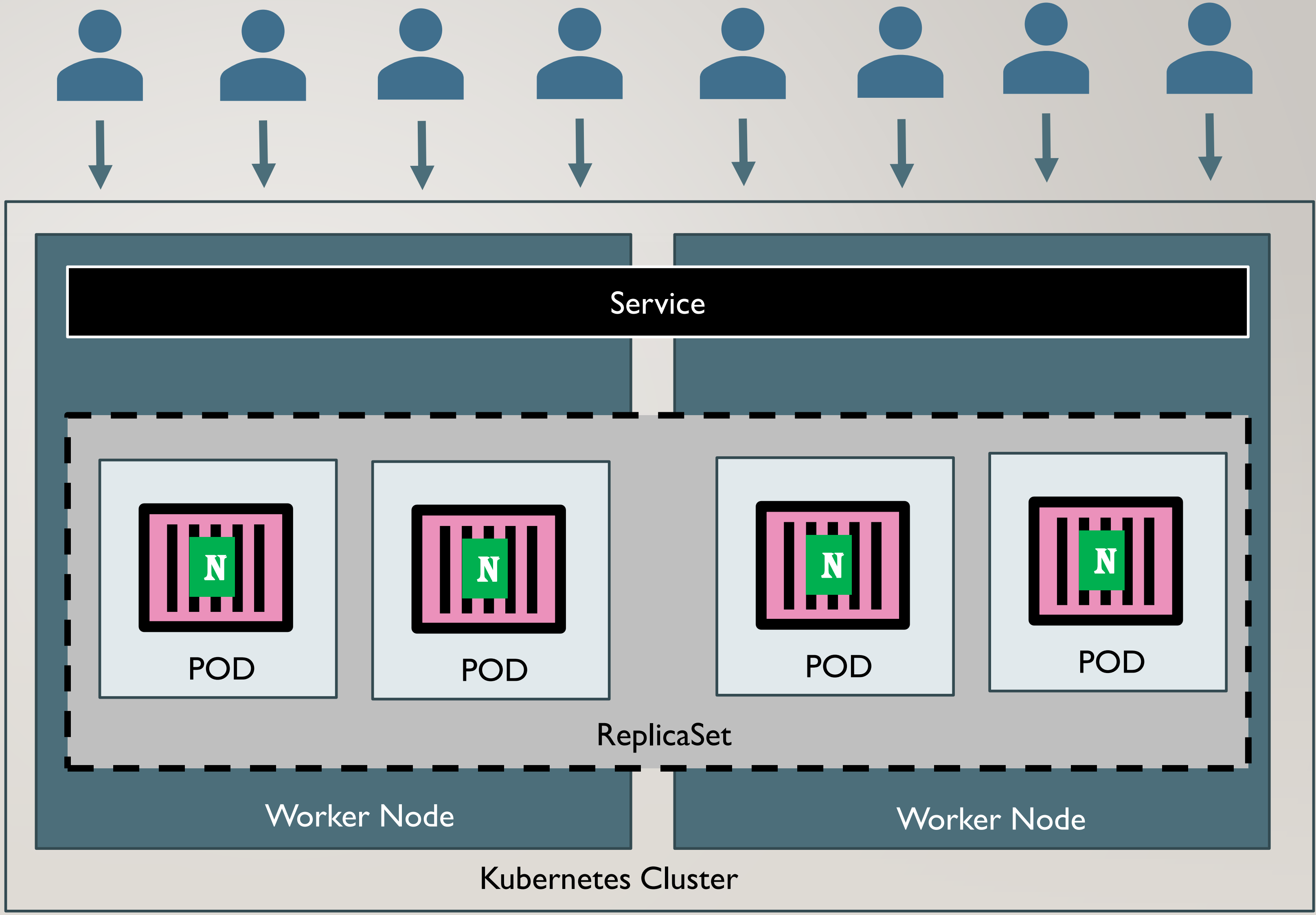
# KUBERNETES – REPLICASET

- Load Balancing
- To avoid overloading of traffic to single pod we can use load balancing.
- Kubernetes provides pod load balancing out of the box using Services  for the pods which are part of a ReplicaSet
- Labels & Selectors are the key items which ties all 3 together (Pod, ReplicaSet & Service), we will know in detail when we are writing YAML manifests for these objects



# KUBERNETES – REPLICASET

- Scaling
- When load become too much for the number of existing pods, Kubernetes enables us to easily scale up our application, adding additional pods as needed.
- This is going to be seamless and super quick.



# Kubernetes

# ReplicaSets

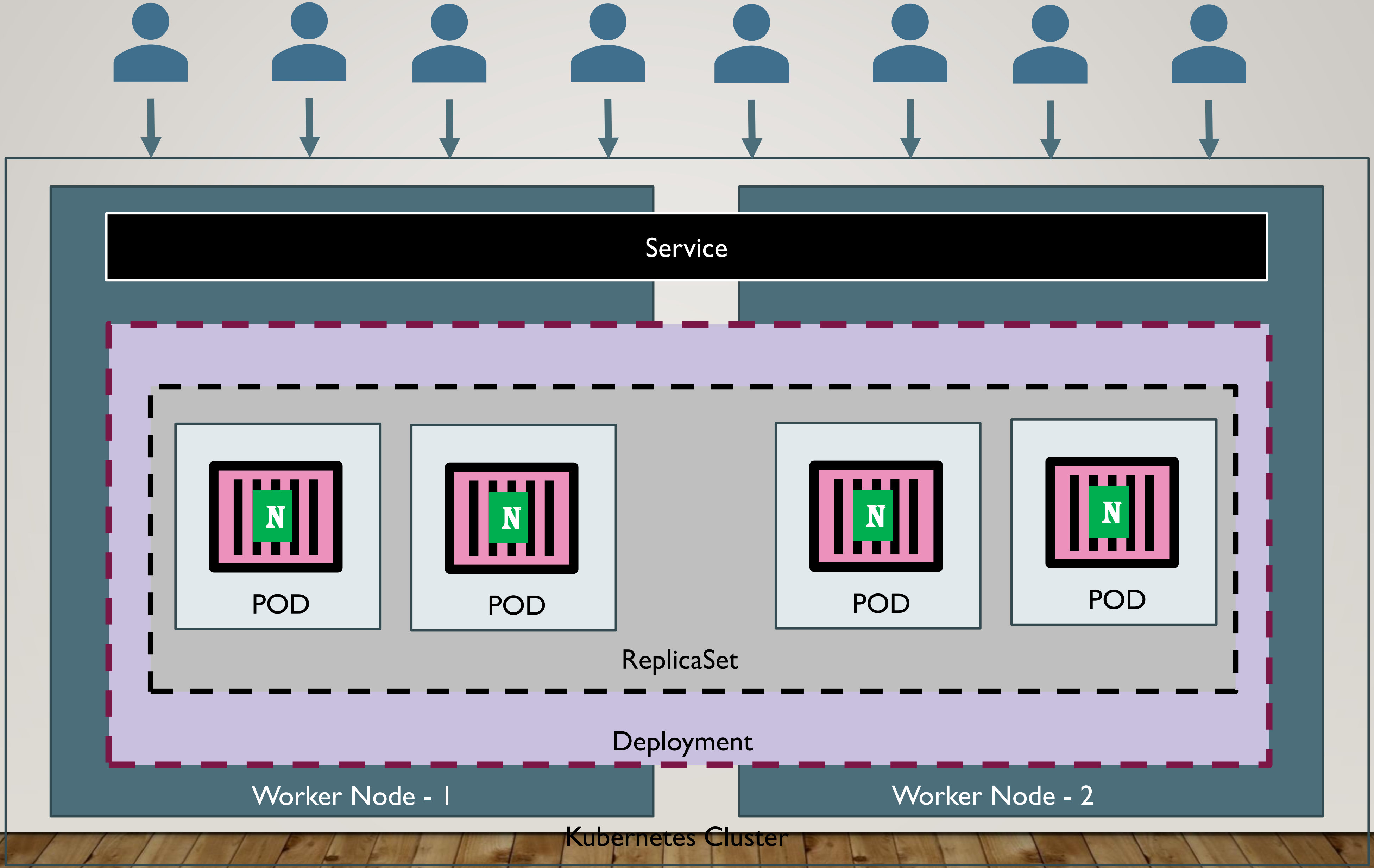
# Demo

# Kubernetes

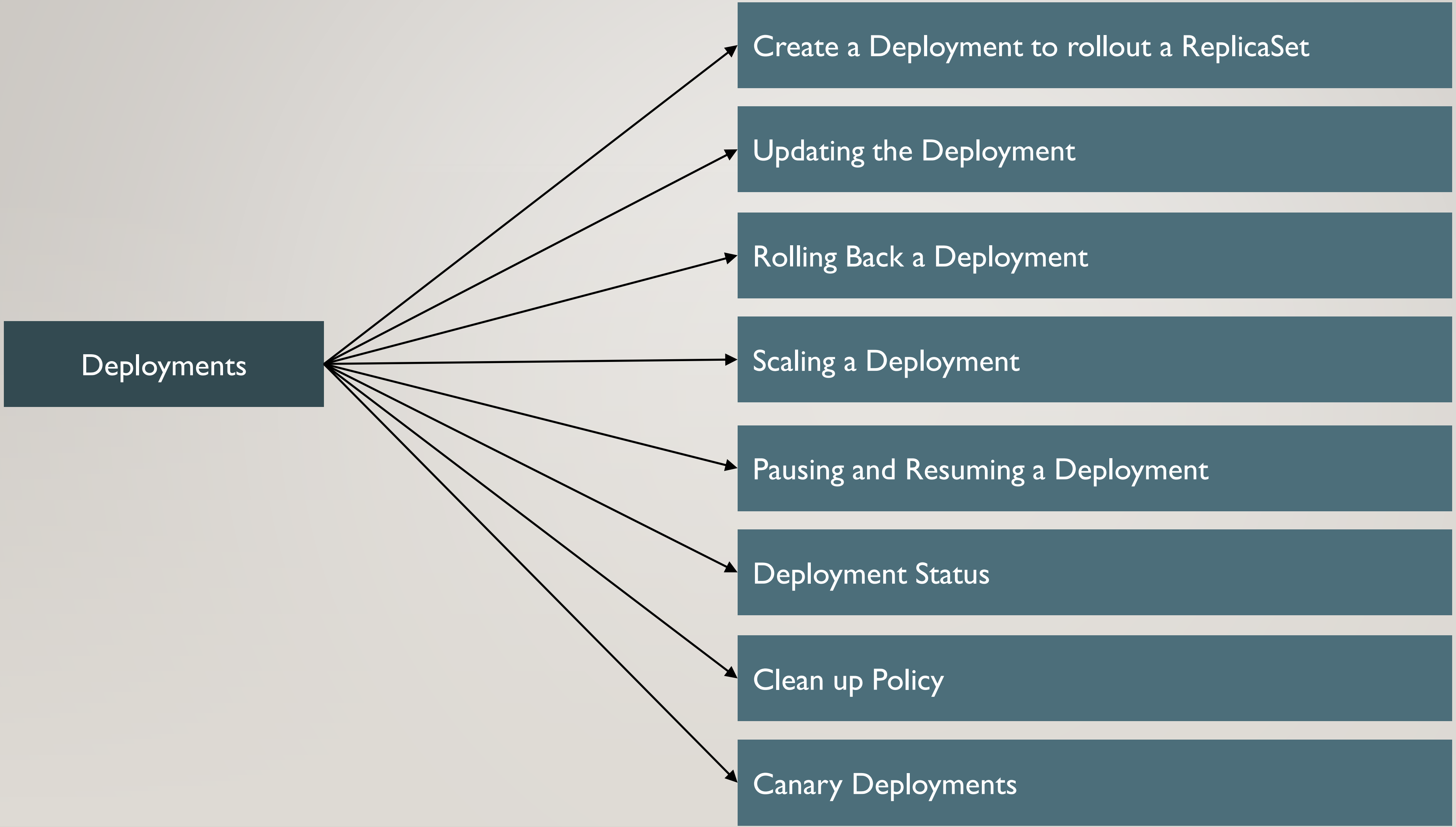
# Deployments

# Kubernetes
# Deployments

# KUBERNETES – DEPLOYMENTS



# KUBERNETES - DEPLOYMENT



Deployments

Create a Deployment to rollout a ReplicaSet

Updating the Deployment

Rolling Back a Deployment

Scaling a Deployment

Pausing and Resuming a Deployment

Deployment Status

Clean up Policy

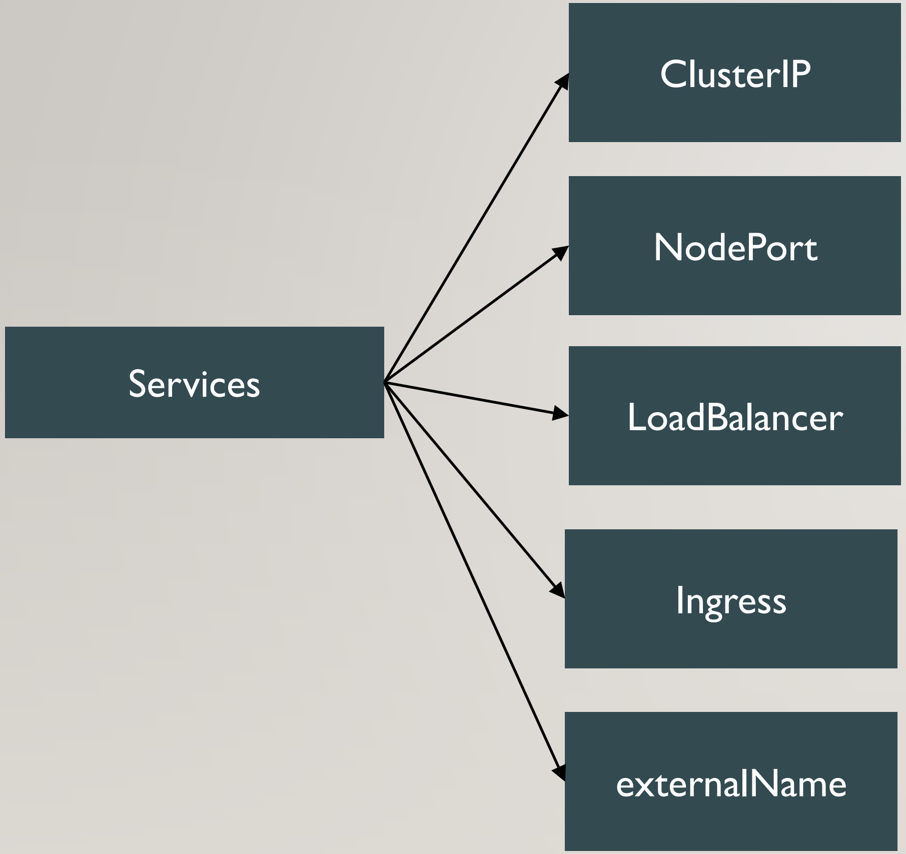
Canary Deployments

# Kubernetes Deployments Demo

# Kubernetes

## Services

# KUBERNETES - SERVICES



Services

ClusterIP

Used for communication between applications inside k8s cluster (Example: Frontend application accessing backend application)

NodePort

Used for accessing applications outside of of k8s cluster using Worker Node Ports (Example: Accessing Frontend application on browser)

LoadBalancer

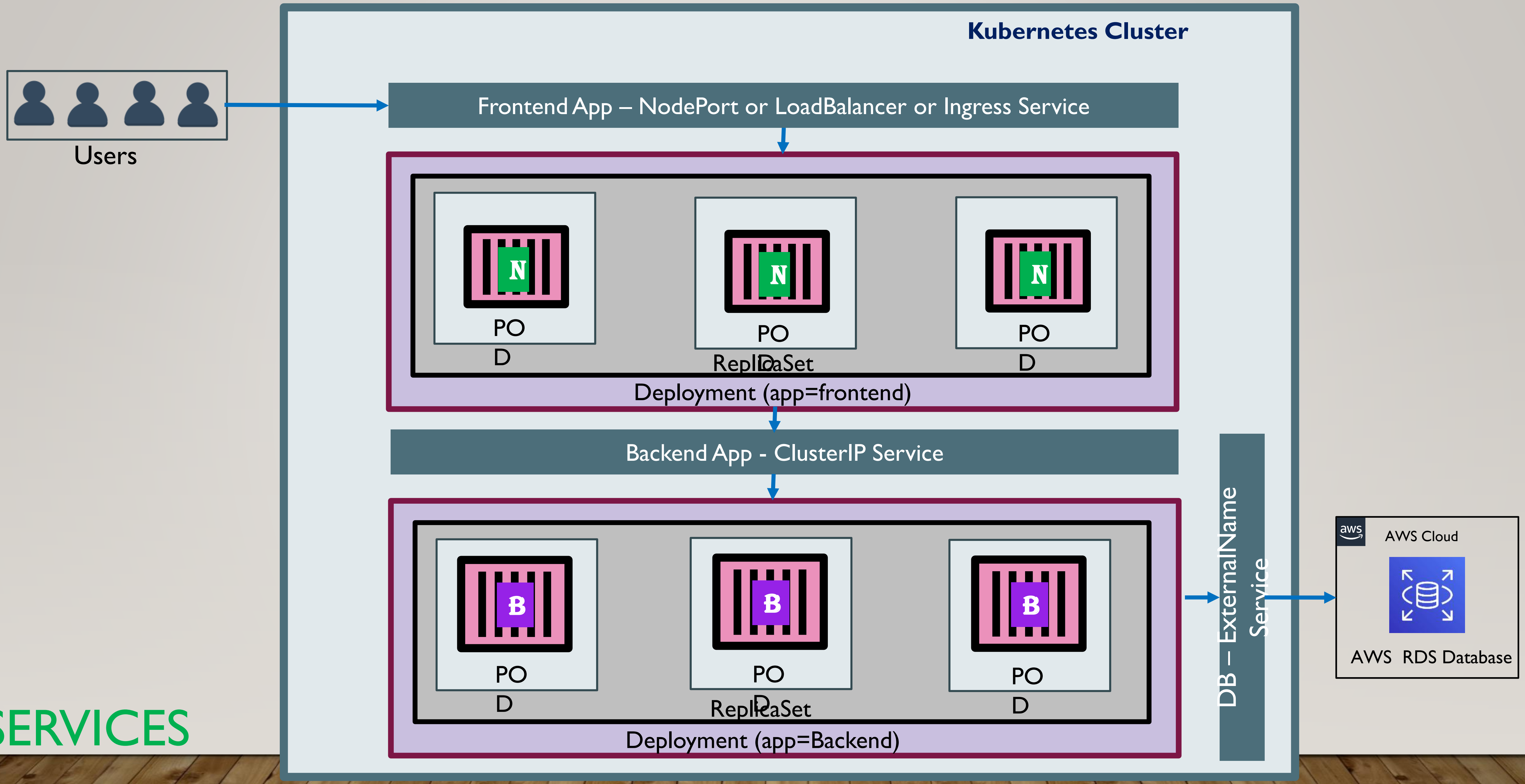
Primarily for Cloud Providers to integrate with their Load Balancer services (Example: AWS Elastic Load Balancer)

Ingress

Ingress is an advanced load balancer which provides Context path based routing, SSL, SSL Redirect and many more (Example: AWS ALB)

externalName

To access externally hosted apps in k8s cluster (Example: Access AWS RDS Database endpoint by application present inside k8s cluster)



Kubernetes Cluster
Users
Frontend App – NodePort or LoadBalancer or Ingress Service
N
PO
D
ReplicaSet
Deployment (app=frontend)
Backend App - ClusterIP Service
B
Deployment (app=Backend)
DB – ExternalName Service
aws
AWS Cloud
AWS RDS Database

# SERVICES

# Kubernetes
# Services
# Demo

# Kubernetes

# YAML Basics

# YAML BASICS

- YAML is not a Markup Language
- YAML is used to store information about different things
- We can use YAML  to define key, Value pairs like variables, lists and objects
- YAML is very similar to JSON (Javascript Object Notation)
- YAML primarily focuses on readability and user friendliness
- YAML is designed to be clean and easy to read
- We can define YAML files with two different extensions
  - abc.yml
  - abc.yaml

# YAML BASICS

- YAML Comments
- YAML Key Value Pairs
- YAML Dictionary or Map
- YAML Array / Lists
- YAML Spaces
- YAML Document Separator